Tensors and deep architectures

I. V. Oseledets,

Skolkovo Institute of Science and Technology, Moscow, Russia

5 June 2016

Skoltech

Сколковский институт науки и технологий

Question

What is the connection between tensor decompositions and deep neural networks?

Disclaimer

There are no tensors (yet!) it TensorFlow.

There is a plan to fix it :)

What is a tensor

A tensor is a d-way array

$$A(i_1, \dots, i_d)$$

and it can be huge: it is described on $\mathcal{O}(n^d)$ parameters,

Which can not be stored for even medium n and d.

Thus, compression is needed

Loss surfaces of multilayer networks

In the paper Choromanska et. al (2014), The loss surfaces of multilayer networks, arxiv:1412.0233

A connection between DNN and *spin-glass models* was established.

Spin-glass

Functional to be minimized:

$$Y = q\sigma\left(W_H^\top \sigma\left(W_{H-1}^\top \ldots \sigma\left(W_1^\top X\right)\right)\ldots\right).$$

- Consider ReLU activation
- Assume randomness of X

Spin-glass

Functional to be minimized:

$$Y = q\sigma\left(W_H^\top \sigma\left(W_{H-1}^\top \ldots \sigma\left(W_1^\top X\right)\right)\ldots\right).$$

- Consider ReLU activation
- Assume randomness of X

Then, the loss surface with respect to weights can be approximated as

Spin-glass

$$\mathcal{L}(\widehat{w}) = C \sum_{i_1,\ldots,i_H} X_{i_1,\ldots,i_H} \widehat{w}_{i_1} \ldots \widehat{w}_{i_H}.$$

This is a polylinear loss function typical in tensor decompositions (it corresponds to best rank-r approximation).

The optimization is non-convex, but has a lot of structure.

Exponential machines

Of the ideas how it can be used was recently proposed in Novikov, Trofimov, Oseledets, Tensor Train polynomial models via Riemannian optimization, arxiv:1605.03795

3-dim example:

$$\begin{aligned}\widehat{y}(\vec{x}) = \mathcal{W}_{000} &+ \mathcal{W}_{100}\, x_1 + \mathcal{W}_{010}\, x_2 + \mathcal{W}_{001} x_3 \\ &+ \mathcal{W}_{110}\, x_1 x_2 + \mathcal{W}_{101}\, x_1 x_3 + \mathcal{W}_{011}\, x_2 x_3 \\ &+ \mathcal{W}_{111}\, x_1 x_2 x_3.\end{aligned} \tag{1}$$

Exponential machines: general form

$$\widehat{y}(\vec{x}) = \sum_{i_1=0}^{1} \ldots \sum_{i_d=0}^{1} \mathcal{W}_{i_1 \ldots i_d} \prod_{k=1}^{d} x_k^{i_k}. \tag{2}$$

The regression will be parametrizes by a tensor with 2^d entries, which will obviously overfit if no structure is assumed.

Generalized tensor

Convolutional Rectifier Networks as Generalized Tensor Decompositions, arxiv: 1603.00162, Choen-Shashua.

Statement: A score generated by as shallow ConvNet is given as

$$A(h_y^S) = \sum_{s=1}^{Z} a_z^y (Fa^{z,1}) \otimes ... \otimes (Fa^{z,N}),$$

where

$$\mathcal{A}(h_y)_{d_1,...,d_N} = h_y(x^{(d_1)}, ..., x^{(d_N)}).$$

It basically encoded the function of N variables.

Classical tensor decomposition

The classical tensor decomposition is the CP (canonical polyadic) decomposition of the form

$$A(i_1, \dots, i_d) \approx \sum_{\alpha=1}^{r} U_1(i_1, \alpha) \dots U_d(i_d, \alpha),$$

and for $d = 2$ it boils down to rank-r approximation, which can be done by the SVD.

Classical tensor decomposition

The classical tensor decomposition is the CP (canonical polyadic) decomposition of the form

$$A(i_1, \ldots, i_d) \approx \sum_{\alpha=1}^{r} U_1(i_1, \alpha) \ldots U_d(i_d, \alpha),$$

and for $d = 2$ it boils down to rank-r approximation, which can be done by the SVD.

The loss surface of the CP can be better than the loss surface of backpropagation (see papers by Anima Anandkumar), but still there can be huge problems!

Problems with CP decomposition

- Best approximation may not exist
- Thus, swap convergence of the algorithms or convergence to the local minima.

Tensor train decomposition

These problems are solved with another polylinear model, named tensor train (TT) decomposition (Oseledets, 2009), also known as matrix product states for more than 50 years in physics.

$$A(i_1, \dots, i_d) \approx G_1(i_1) \dots G_d(i_d),$$

where $G_k(i_k)$ is a matrix of size $r_{k-1} \times r_k$ for any fixed i_k, and $r_0 = r_d = 1$.

TT-decomposition

$$A(i_1, \dots, i_d) \approx G_1(i_1) \dots G_d(i_d),$$

It has the properties of multilinear SVD (similar algorithms, low complexity if the TT-ranks are small).

- It defines a manifold in a d-dim space
- It is easy to optimize over such manifold in the framework of Riemannian optimization, and the algorithms converge very well (still under investigation).

Tensor-train: basic facts

We can do a lot in the TT-format

- Computation of the quasi-optimal approximation by TT-SVD
- Optimal recovery from sampling (TT-cross approximation)
- There is an open-source software for working with tensors in the TT-format (TT-Toolbox).

Optimization over TT-manifolds

A simple approach is *alternating least squares*.

Update one G_k at a time, since it is polylinear, the optimization is cheap.

For non-quadratic optimizations, Riemannian optimization is a method of choice.

Riemannian optimization: problem setting

Illustrate for $d = 2$, matrices of rank-r:

$$X = USV^\top,$$

where U and V have r orthonormal columns, S is $r \times r$.

We need to minimize

$$F(X) \to \min, \quad X \in \mathcal{M}_r.$$

Riemannian optimization(2)

$$F(X) \to \min, \quad X \in \mathcal{M}_r.$$

where

$$X = USV^\top.$$

We can differentiate with respect to U, S and V by it is not a good idea, because it is overparametrized:

$$UV^\top = U\Phi\Phi^{-1}V^\top$$

for any Φ.

Riemannian optimization (3)

Instead, we make a step in X:

$$Y_{k+1} = X_k + \alpha \nabla F(X_k),$$

and project it onto the tangent space:

$$X_{k+1} = P_{\mathcal{T}}(Y_{k+1}).$$

We are not yet on the manifold, so we need to do retraction onto the manifold,

$$X_{k+1} = R(X_k + \alpha P_{\mathcal{T}}(\nabla F(X_k).$$

Using Riemannian optimization

It all applies with technical changes to TT-format (but not to the CP-format).

It has been applied:

- Matrix completion
- Exponential machines
- Several papers on NIPS/ICML use it.

Exponential machines: results

$$\widehat{y}(\vec{x}) = \sum_{i_1=0}^{1} \dots \sum_{i_d=0}^{1} \mathcal{W}_{i_1 \dots i_d} \prod_{k=1}^{d} x_k^{i_k}. \tag{3}$$

Here we introduce a TT-regularization: the weight tensor $\mathcal{W}$ is restricted to the case of rank-r tensors and updated via stochastic Riemannian approach.

Results (synthetic data with high-order interaction)

`https://github.com/bihaqo/exp-machines`

Method	Test AUC	Training time (s)	Inference time (s)
Log. reg.	0.50 ± 0.0	0.4	0.0
RF	0.55 ± 0.0	21.4	1.3
SVM RBF	0.50 ± 0.0	2262.6	1076.1
SVM poly. 2	0.50 ± 0.0	1152.6	852.0
SVM poly. 6	0.56 ± 0.0	4090.9	754.8
2-nd order FM	0.50 ± 0.0	638.2	0.1
6-th order FM	0.57 ± 0.05	1412.0	0.2
ExM rank 2	0.54 ± 0.05	198.4	0.1
ExM rank 4	0.69 ± 0.02	443.0	0.1

Example of tensors in DNN

One of the main applications is the layer compression

Many layers of a DNN are in fact contractions with 3D, 4D, ... tensors.

Compression of a conv. layer

Speeding up convolutional neural network using fine-tuned CP-decomposition, Lebedev et. al., ICLR 2015.

In a generalized convolution the kernel tensor is 4D $(d \times d \times S \times T)$ (spatial, input, output).

If we construct rank-R CP-decomposition, that amounts to having two layers of smaller total complexity, than the full layer.

The idea: use TensorLab (best MATLAB code for CP-decomposition) to initialize these two layers, and then fine-tune

Result: 8.5x speedup with 1% accuracy drop.

TensorNet

Novikov et. al, NIPS 2015: use TT-structured layer instead of a dense layer.

Gives a huge compression!

Tensor networks

Tensor-train is a special case of linear tensor network:

$$A(i_1, \dots, i_d) \approx \sum_{\alpha_1, \dots, \alpha_{d-1}} G_1(i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \dots G_d(\alpha_{d-$$

In general, there is a graph, where vertices correspond to original indices, and edges - to the summation indices (similar to MRF).

If the graph has loops, than we have problems

Although, in particular cases we can still optimize

Algorithms for tensor networks

In physics, the algorithm for the optimization of tensor networks are based on the so-called renormalization group.

The idea is to do local optimization while fixing the environment.

It might be the alternative/enhancement of backpropagation.

Future work

- Tensors are very efficient for PDEs/integral equations
- DNN are not (not many people are working in this area), and if we can put tensors on TensorFlow there can be a huge breakthough in modelling.

Conclusions

- Tensor factorizations as compression tools are already there
- Allow for efficient optimizations with few iterations
- Framework of Riemannian optimization is great

Links

- oseledets.github.io – group webpage
- i.oseledets@skoltech.ru – email
- http://github.com/oseledets – projects